Hybrid Eulerian-Lagrangian Semi-Implicit Time-Integrators

Francis X. Giraldo
Naval Research Laboratory, Monterey, CA 93943, USA
email: giraldo@nrlmry.navy.mil

Abstract. Hybrid Eulerian-Lagrangian semi-implicit time-integrators (HELSI) are presented which use the standard semi-implicit formulation as their starting point. The advantage of such an approach is that existing models which employ Eulerian semi-implicit time-integrators can easily be augmented to use the HELSI approach, that is, provided that either: a series of advection problems can be solved efficiently as in the operator-integration-factor splitting (OIFS) method or that the grid and underlying numerics allow the construction of efficient search and interpolation stencils typically required of semi-Lagrangian methods. Once the advection problem is solved the approach then follows the standard semi-implicit approach. An important caveat of this approach is that the semi-implicit time-integrator must be constructed around the implicit backward difference formulas.

Key Words backward difference formula, operator-integration factor splitting, shallow water equations, semi-implicit, semi-Lagrangian, spectral element method

1 Introduction

Although explicit time-integrators are the easiest methods to implement their main disadvantage is that small time steps must be observed in order to maintain stability. In atmospheric modeling, the reason for this prohibitively small time-step is due to the fast moving gravity waves. These waves require a small time-step while only carrying a very small percent of the total energy in the system. In order to ameliorate this rather stringent time-step restriction researchers have tried various approaches such as using a larger differencing stencil for the gravity wave terms thereby effectively reducing the Courant number [1, 2], and discretizing the gravity wave terms implicitly in time [3]. It turns out that discretizing the gravity wave terms implicitly in time is the more effective way of increasing the time-step.

After the gravity wave terms have been successfully discretized the next set of terms responsible for controlling the maximum time-step are the Rossby waves (advection terms). In order to use increasingly larger time steps researchers have turned to Lagrangian methods for treating these recalcitrant terms [4, 5]. By rewriting the equations in terms of the Lagrangian derivative the troublesome advection terms are absorbed into the Lagrangian derivative. Thus the equations in this form are now discretized in time along characteristics which results in a much more stable numerical method due to the virtual disappearance of the Courant-Friedrichs-Lewy (CFL) condition.

In the current paper we follow the approach put forth by Maday et al. [6] which they called operator-integration-factor splitting or OIFS for short. The approach that we present in the current work uses the idea of an integration factor and the OIFS approach to present a unified view of semi-Lagrangian and OIFS methods. The idea of viewing SL and OIFS as similar methods is not new. In fact, Maday et al. [6] showed that in the OIFS method the advection integrating factor is an approximation to the Lagrangian derivative while Boyd [7] refers to SL as an integration factor method and Xiu et al. [8] compare the two methods for the incompressible Navier-Stokes equations. Two recently proposed approaches for the shallow water equations (SWE) that deserve mention are the high-order SL method of Giraldo et al. [9] and the OIFS method of St-Cyr and Thomas [10]. Although these two approaches have shown to be quite powerful, they are particularly geared

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Hybrid Eulerian-Lagrangian Semi-Implicit Time-Integrators	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
International Journal of Computers and Mathematics with Applications, Vol. 52 pp. 1325-1342 (2006)

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**18**	

for the SWE. For this reason, it is not immediately obvious how to extend these two methods for more complicated equation sets such as the hydrostatic primitive equations (HPE) used in numerical weather and climate modeling or the oceanic Navier-Stokes equations.

In short, the contribution of the current manuscript to the literature is that we show how to build a unified view of integration factors directly onto the standard semi-implicit method. This makes it quite straightforward to augment an existing semi-implicit model including highly complex models that solve the HPE which we reserve for a forthcoming paper.

2 Shallow Water Equations

While the proposed time-integrators that we refer to as HELSI can be used on any time-dependent system of partial differential equations (PDEs), to simplify the exposition we shall apply them only to the shallow water equations (SWE) on a rotating sphere. The reason for choosing this set of PDEs is because the spherical domain allows boundary conditions to be treated quite simply using periodicity and because the SWE are a good testbed for the construction of atmospheric models and thereby represent a relevant equation set.

The shallow water equations are a system of first order nonlinear hyperbolic equations which govern the motion of an inviscid incompressible fluid in a shallow depth. The predominant feature of this type of fluid is that the characteristic length of the fluid is far greater than its depth which is analogous to the motion of air in the atmosphere and water in the ocean. For this reason these equations are typically used as a first step towards the construction of numerical weather prediction (NWP), climate, and ocean models.

Let us write the SWE in the following vector form

$$\frac{\partial \boldsymbol{q}}{\partial t} = \boldsymbol{S}_E(\boldsymbol{q}) \tag{1}$$

with

$$S_E(\boldsymbol{q}) = -\begin{pmatrix} \boldsymbol{\nabla}\cdot(\phi\boldsymbol{u}) \\ \boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u} + \frac{2\Omega z}{a^2}(\boldsymbol{x}\times\boldsymbol{u}) + \boldsymbol{\nabla}(\phi+\phi^s) \end{pmatrix} \tag{2}$$

where the subscript E is meant to remind the reader that the equations are written in an Eulerian reference frame. Furthermore

$$\boldsymbol{q} = (\phi, u, v, w)^T$$

represents the state vector composed of the geopotential height, ϕ, and the three Cartesian velocity components, (u, v, w), and ϕ^s represents the surface topography (e.g., mountains).

Alternatively, the SWE can be written more compactly in terms of the Lagrangian derivative

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \boldsymbol{u}\cdot\boldsymbol{\nabla} \tag{3}$$

as follows

$$\frac{d\boldsymbol{q}}{dt} = \boldsymbol{S}(\boldsymbol{q}) \tag{4}$$

where the source function is now defined as

$$S(\boldsymbol{q}) = -\begin{pmatrix} \phi\boldsymbol{\nabla}\cdot\boldsymbol{u} \\ \frac{2\Omega z}{a^2}(\boldsymbol{x}\times\boldsymbol{u}) + \boldsymbol{\nabla}(\phi+\phi^s) \end{pmatrix}. \tag{5}$$

3 Eulerian Semi-Implicit Time-Integrators

The Eulerian semi-implicit (ESI) discretization of Eq. (1) is as follows

$$\frac{\partial \boldsymbol{q}}{\partial t} = \{S_E(\boldsymbol{q}) - \delta L(\boldsymbol{q})\} + \delta\,[L(\boldsymbol{q})] \tag{6}$$

where the terms inside the curly brackets are time-integrated explicitly, those inside the square brackets implicitly, L represents the linearization of S, and $\delta = 0$ or 1 depending on whether a purely explicit or semi-implicit method is chosen. The linearized operator L is in fact defined as follows

$$L(\boldsymbol{q}) = -\begin{pmatrix} \Phi \boldsymbol{\nabla} \cdot \boldsymbol{u} \\ \boldsymbol{\nabla} \phi \end{pmatrix} \tag{7}$$

where Φ denotes some mean height. It should be mentioned that the operator L does not change from the Eulerian to Lagrangian reference frame because no linearization about the velocity field is applied consistently with the approach undertaken for all atmospheric models that use the hydrostatic primitive equations.

In [11] we studied various Eulerian semi-implicit time-integrators and determined that for the atmospheric hydrostatic primitive equations (HPE) the second order backward difference formulas (BDF2) are a good choice. A straightforward BDF2 discretization of Eq. (6) for $\delta = 0$ is

$$\boldsymbol{q}^{n+1} = \alpha_0 \boldsymbol{q}^n + \alpha_1 \boldsymbol{q}^{n-1} + \gamma \Delta t S_E(\boldsymbol{q})^{n+1}. \tag{8}$$

However, Eq. (8) would require solving an implicit nonlinear problem. Instead, we use the splitting proposed by Karniadakis et al. [12] to extrapolate the nonlinear terms in $S_E(\boldsymbol{q})^{n+1}$ with known values and solving the linear terms implicitly; this is the idea behind the BDF2 semi-implicit time-integrator which we now describe.

A BDF2 semi-implicit time-integration of Eq. (1) is then

$$\boldsymbol{q}^{n+1} = \sum_{m=0}^{1} \alpha_m \boldsymbol{q}^{n-m} + \gamma \Delta t \sum_{m=0}^{2} \beta_m S_E(\boldsymbol{q})^{n-m} + \delta \gamma \Delta t L \left(\sum_{m=-1}^{2} \rho_m \boldsymbol{q}^{n-m} \right) \tag{9}$$

where the βs are the extrapolation coefficients and the term inside L is obtained by combining the explicit and implict L in Eq. (6). For completeness in Table I we include the BDF2 coefficients; note that the scheme BDF2A is the semi-implicit time-integrator proposed in [12] and BDF2B was proposed by Hulsten (personal communication) and used for the hydrostatic primitive equations in [11]. For all numerical experiments we use BDF2B exclusively (the stability regions of BDF2A and BDF2B can be found in [11]).

Method	α_0	α_1	γ	β_0	β_1	β_2	ρ_{-1}	ρ_0	ρ_1	ρ_2
BDF2A	4/3	-1/3	2/3	2	-1	0	1	-2	1	0
BDF2B	4/3	-1/3	2/3	8/3	-7/3	2/3	1	-8/3	7/3	-2/3

Table I: Coefficients of the backward difference formulas corresponding to Eq. (9).

The reason why Eq. (9) is described as an Eulerian semi-implicit time-integrator is because all of the quantities present are constructed in an Eulerian fashion, that is, $\boldsymbol{q}$ represents values at a specific grid point but defined at the various time-levels ranging from $n-1$, n, to $n+1$. Note that $\boldsymbol{q}$ in the summation term $\sum_{m=0}^{1} \alpha_m \boldsymbol{q}^{n-m}$ is constructed in an explicit Eulerian fashion. This point will become much more apparent in the next section where we discuss the HELSI time-integrators.

We can now simplify Eq. (9) by extracting its fully explicit solution as follows

$$\boldsymbol{q}_E^{\text{explicit}} = \sum_{m=0}^{1} \alpha_m \boldsymbol{q}^{n-m} + \gamma \Delta t \sum_{m=0}^{2} \beta_m S_E(\boldsymbol{q})^{n-m}. \tag{10}$$

Multiplying Eq. 9 by ρ_{-1}, and adding $\sum_{m=0}^{2} \rho_m \boldsymbol{q}^{n-m}$ yields

$$\boldsymbol{q}_{tt} = \widehat{\boldsymbol{q}}_E + \delta \gamma \Delta t \rho_{-1} L(\boldsymbol{q}_{tt}) \tag{11}$$

where

$$\widehat{\boldsymbol{q}}_E = \rho_{-1} \boldsymbol{q}_E^{\text{explicit}} + \sum_{m=0}^{2} \rho_m \boldsymbol{q}^{n-m} \tag{12}$$

and

$$\boldsymbol{q}_{tt} = \sum_{m=-1}^{2} \rho_m \boldsymbol{q}^{n-m} \equiv \rho_{-1}\boldsymbol{q}^{n+1} + \sum_{m=0}^{2} \rho_m \boldsymbol{q}^{n-m}. \tag{13}$$

Eq. (11) is the actual expression used to solve the semi-implicit problem. This completes the construction of the Eulerian semi-implicit time-integrator.

4 Hybrid Eulerian-Lagrangian Semi-Implicit Time-Integrators

4.1 Review of Operator Integration Factors

In order to understand the unified view of OIFS and SL it is important to understand the concept of integrating factors; this idea is used for simplifying the solution of partial differential equations (PDEs) by essentially eliminating one of the terms. Let us write a generalized hyperbolic-parabolic PDE in the following way

$$\frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{q} = \boldsymbol{S} \tag{14}$$

where $\boldsymbol{S}$ is a vector that contains all the source terms. We can now define an integrating factor, $\boldsymbol{Q}^{t^*}(t) \in R^{N \times N}$, to simplify the original PDE to the following form

$$\frac{\partial}{\partial t}\boldsymbol{Q}^{t^*}(t) \cdot \boldsymbol{q}(t) = \boldsymbol{Q}^{t^*}(t) \cdot \boldsymbol{S} \tag{15}$$

where N is the number of grid points, $t^* \geq t$ is any time at which we would like to evaluate the integrating factor, and $\boldsymbol{Q}^{t^*}(t^*) = I$ where $I \in R^{N \times N}$ is the identity matrix. It turns out (see [6]) that the following equality can be derived

$$\boldsymbol{Q}^{t^*}(t) \cdot \boldsymbol{q}(t) = \widetilde{\boldsymbol{q}}^{(t^*;t)}(t^* - t) \tag{16}$$

which then allows us to rewrite Eq. (15) as follows

$$\frac{\partial}{\partial t}\widetilde{\boldsymbol{q}}^{(t^*;t)}(t^* - t) = \boldsymbol{Q}^{t^*}(t) \cdot \boldsymbol{S} \tag{17}$$

where the new variable $\widetilde{\boldsymbol{q}}$ is the solution to the advection problem

$$\frac{\partial}{\partial s}\widetilde{\boldsymbol{q}}^{(t^*;t)}(s) = -\boldsymbol{u}(s) \cdot \boldsymbol{\nabla}\widetilde{\boldsymbol{q}}^{(t^*;t)}(s) \tag{18}$$

for $0 < s < t* - t$ with the initial condition $\widetilde{\boldsymbol{q}}^{(t^*,t)}(0) = \boldsymbol{q}(t)$. Now, setting $t^* = t^{n+1}$ yields the final form of Eq. (17)

$$\frac{\partial}{\partial t}\widetilde{\boldsymbol{q}}^{(t^{n+1};t^n)}(\Delta t) = \boldsymbol{S} \tag{19}$$

where the solution of Eq. (18) can be carried out in any desired way. We shall next explore using the semi-Lagrangian (SL) and Operator-Integration-Factor Splitting (OIFS) methods. Comparing Eqs. (14) and (19) it becomes immediately obvious that we could have written the original system as follows

$$\frac{d\boldsymbol{q}}{dt} = \boldsymbol{S} \tag{20}$$

which is the form we shall use in the next section.

4.2 HELSI

The hybrid Eulerian-Lagrangian semi-implicit discretization of Eq. (4) is

$$\frac{d\boldsymbol{q}}{dt} = \{S(\boldsymbol{q}) - \delta L(\boldsymbol{q})\} + \delta\left[L(\boldsymbol{q})\right] \tag{21}$$

where, once again, the terms inside the curly brackets are time-integrated explicitly, and those inside the square brackets implicitly. As mentioned previously the linearized operator L is defined as in the Eulerian case, that is, Eq. (7).

Applying HELSI to Eq. (21) yields

$$\boldsymbol{q}^{n+1} = \sum_{m=0}^{1} \alpha_m \widetilde{\boldsymbol{q}}^{n-m} + \gamma\Delta t \sum_{m=0}^{2} \beta_m S(\boldsymbol{q})^{n-m} + \delta\gamma\Delta t L\left(\sum_{m=-1}^{2} \rho_m \boldsymbol{q}^{n-m}\right) \tag{22}$$

where the tilde above $\boldsymbol{q}$ denotes that this quantity is determined along characteristics in a semi-Lagrangian sense. The reason why Eq. (22) is described as a hybrid Eulerian-Lagrangian semi-implicit time-integrator is because the quantities corresponding to the time derivatives ($\widetilde{\boldsymbol{q}}$) are computed in a Lagrangian sense while the remaining $\boldsymbol{q}$ corresponding to the source function ($\boldsymbol{S}$) and its linearization ($\boldsymbol{L}$) are computed in an Eulerian fashion. Note that by using BDFs the source term does not have to be evaluated along the Lagrangian path (unlike in semi-Lagrangian centered schemes typically used in NWP models [4, 5]). This allows for a standard semi-implicit model to be easily augmented to a Lagrangian semi-implicit model; furthermore, because the source terms are not computed along trajectories allows the model to retain accuracy and conservation.

Applying the exact same simplifications that were used in ESI we can now write the final HELSI form as follows

$$\boldsymbol{q}_{tt} = \widehat{\boldsymbol{q}} + \delta\gamma\Delta t\rho_{-1} L\left(\boldsymbol{q}_{tt}\right) \tag{23}$$

where

$$\widehat{\boldsymbol{q}} = \rho_{-1}\boldsymbol{q}^{\text{explicit}} + \sum_{m=0}^{2} \rho_m \boldsymbol{q}^{n-m} \tag{24}$$

and

$$\boldsymbol{q}^{\text{explicit}} = \sum_{m=0}^{1} \alpha_m \widetilde{\boldsymbol{q}}^{n-m} + \gamma\Delta t \sum_{m=0}^{2} \beta_m S(\boldsymbol{q})^{n-m}. \tag{25}$$

Comparing Eqs. (9) and (22) we see that the only change required to augment an ESI to a HELSI is to subtract the advection terms from the source function $\boldsymbol{S}_E$ and then deciding how $\widetilde{\boldsymbol{q}}$ will be approximated along the characteristics. To reiterate, what makes this augmentation so straightforward is the BDF-based construction of the semi-implicit method. In addition, Eqs. (23) through (25) show that SL and OIFS methods may be managed in the same code simply by replacing $\widetilde{\boldsymbol{q}}$ with the proper approximation. In fact, our numerical results were obtained by virtue of a single code. Let us now describe how to solve the advection step in a Lagrangian sense along characteristics.

4.3 Advection Step

To complete the HELSI discretization requires the construction of $\widetilde{\boldsymbol{q}}$ which are the solution values along the characteristics. At this point we have a choice as to which method to use for this step. The choices are numerous but the candidates that we shall explore in this work are the operation-integration-factor splitting (OIFS) and semi-Lagrangian (SL) methods (another choice used previously is the characteristic Galerkin [13]). Regardless of which method is selected the goal is to solve the Lagrangian derivative

$$\frac{d\boldsymbol{q}}{dt} = 0 \tag{26}$$

accurately and efficiently.

4.3.1 Semi-Lagrangian Method

In the semi-Lagrangian (SL) method one simply replaces Eq. (26) by the solution $\boldsymbol{q}^{n+1} = \boldsymbol{q}(\widetilde{\boldsymbol{x}}^n, t^n)$ where $\widetilde{\boldsymbol{x}}^n = \boldsymbol{x}(t^n)$ is the foot of the characteristics that departs from $\boldsymbol{x}(t^n)$ and arrives at $\boldsymbol{x}(t^{n+1})$. Thus the advection step with the semi-Lagrangian method reduces to computing the departure point via the particle trajectory equation

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}. \tag{27}$$

There are various ways of solving Eq. (27) but typically it is carried out via the implicit midpoint rule (which requires iterations) or by Runge-Kutta (RK) methods. For example, using the second order Runge-Kutta (RK2) method yields

$$\begin{aligned} \widetilde{\boldsymbol{x}}^{n+1/2} &= \boldsymbol{x}^{n+1} - \frac{\Delta t}{2}\boldsymbol{u}^{n+1} \\ \widetilde{\boldsymbol{x}}^{n} &= \boldsymbol{x}^{n+1} - \Delta t\widetilde{\boldsymbol{u}}^{n+1/2} \end{aligned} \tag{28}$$

where $\widetilde{\boldsymbol{x}}^n = \boldsymbol{x}(t^n)$ and $\widetilde{\boldsymbol{u}}^{n+1/2} = \boldsymbol{u}(x(t^{n+1/2}), t^{n+1/2})$. Because the velocity field is known only at integer time-levels (e.g., n-1, n, n+1) then to get the velocity at intermediate times (e.g., n-1/2 and n+1/2) requires interpolation and extrapolation.

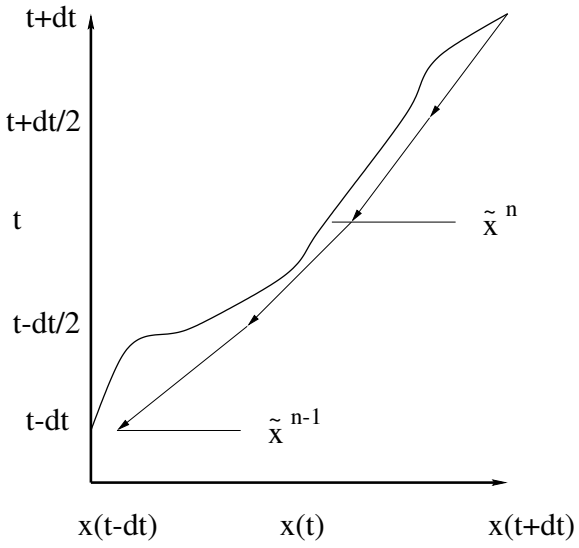


Figure 1: Schematic of the semi-Lagrangian trajectory computation. The thick line is the actual trajectory while the arrows denote the path followed to compute the departure points using a RK2 approximation.

Once the departure point is computed we then have to interpolate to get $\widetilde{\boldsymbol{q}}^n = \boldsymbol{q}(\widetilde{\boldsymbol{x}}^n, t^n)$ because $\widetilde{\boldsymbol{x}}^n$ will not generally fall on a grid point. However looking at Eq. (22) we see that we not only need $\widetilde{\boldsymbol{q}}^n$ but also $\widetilde{\boldsymbol{q}}^{n-1}$. Thus we apply Eq. (28) one more time to get

$$\begin{aligned} \widetilde{\boldsymbol{x}}^{n-1/2} &= \widetilde{\boldsymbol{x}}^{n} - \frac{\Delta t}{2}\widetilde{\boldsymbol{u}}^{n} \\ \widetilde{\boldsymbol{x}}^{n-1} &= \widetilde{\boldsymbol{x}}^{n} - \Delta t\widetilde{\boldsymbol{u}}^{n-1/2} \end{aligned} \tag{29}$$

which is then used to compute $\widetilde{\boldsymbol{q}}^{n-1} = \boldsymbol{q}(\widetilde{\boldsymbol{x}}^{n-1}, t^{n-1})$.

A few things can be immediately discerned about the SL method:

1. The departure point calculation, Eq. (28), is insensitive to the dimension of $\widetilde{\boldsymbol{q}}$.

2. The SL method becomes increasingly more cost-effective with increasing time-step.

3. Because the departure point values, $\widetilde{\boldsymbol{q}}$, are computed via interpolation this then allows monotonicity to be incorporated into the interpolation method. One common practice is to introduce a quasi-flux-corrected transport (FCT) method.

4. Some structure in the grid must exist in order to search efficiently for the departure points $\widetilde{\boldsymbol{x}}$ which are then used to interpolate $\widetilde{\boldsymbol{q}}$.

5. An interpolation stencil must be readily available which requires either structure in the grid, local high-order interpolation functions, or an easily computable polynomial basis construction of the discrete spatial operators.

6. On a distributed-memory computer the departure point of a specific grid point may lie off-processor.

Items 1, 2, and 3 are the virtues of the SL method while items 4, 5, and 6 are its vices. Item 1 is particularly important for systems of equations with multiple solution variables. An example is a transport model which may carry up to 50 tracers.

Since the cost of computing the trajectory and then interpolating is not dependent on the time-step size (item 2) then for pure advection problems the SL method becomes increasingly more cost effective with increasing time-step. However, for more complex systems the condition number of the matrix (resulting from the semi-implicit approach) increases with increasing time-step which will adversely affect the efficiency of the overall solution approach. Item 3 allows monotonicity to be built directly into the interpolation stencil as has been done in [14, 15, 16, 9].

Item 4 says that the limitation of the SL method is that grid structure must exist if efficient search and interpolation operators are to be constructed. In [16, 17, 9] SL formulations on unstructured grids were successfully implemented but constructing efficient search algorithms on unstructured grids for general applications remains an open issue.

Item 5 implies that either there must exist some structure in the grid in order to construct global interpolation functions (e.g., cubic splines) or local high-order interpolation functions must be readily available. High-order interpolation functions are available, for example, if spectral elements are used to approximate the spatial derivatives. This idea was exploited in [16, 17, 9] to construct high-order SL methods on unstructured grids. However, in these works the numerical method used to approximate the spatial derivatives were constructed from an easily computable polynomial expansion (Legendre and Proriol-Koornwinder-Dubiner polynomials); such a polynomial-derived basis expansion may not always be available. An example where non-polynomial bases arise is in the prolate spheroidal wave functions described in [18] and the bubble-type *ad hoc* high order finite elements derived in [19] both of which we have explored in the context of constructing accurate and efficient nonlinear models. For these types of expansion bases OIFS is the only recourse.

Perhaps the worse vice of the SL method (item 6) is that on distributed-memory computers the departure point calculation may require communication across processors (something that should be minimized for the sake of efficiency) because departure points may lie off-processor. Let us next explore a second option for computing the departure point values.

4.3.2 Operator-Integration-Factor Splitting Method

In the operator-integration-factor splitting (OIFS) method [6] the Lagrangian derivative is computed using Eulerian substepping, that is, the goal is to solve

$$\frac{\partial \boldsymbol{q}}{\partial s} = -\breve{\boldsymbol{u}} \cdot \boldsymbol{\nabla} \boldsymbol{q} \tag{30}$$

with $\boldsymbol{q}(s=0) = \boldsymbol{q}^n$ for the approximation of $\widetilde{\boldsymbol{q}}^n$ and $\boldsymbol{q}(s=0) = \boldsymbol{q}^{n-1}$ for computing $\widetilde{\boldsymbol{q}}^{n-1}$. To see how the OIFS method is applied let us assume that we wish to solve Eq. (30) using RK2. We then

get

$$\begin{aligned} \widehat{\boldsymbol{q}}_1^{n+1/2} &= \boldsymbol{q}^n - \frac{\Delta t}{2}\check{\boldsymbol{u}}^n \cdot \boldsymbol{\nabla}\boldsymbol{q}^n \\ \widehat{\boldsymbol{q}}_1^{n+1} &= \boldsymbol{q}^n - \Delta t\check{\boldsymbol{u}}^{n+1/2} \cdot \boldsymbol{\nabla}\widehat{\boldsymbol{q}}_1^{n+1/2} \end{aligned} \tag{31}$$

and

$$\begin{aligned} \widehat{\boldsymbol{q}}_2^{n-1/2} &= \boldsymbol{q}^{n-1} - \frac{\Delta t}{2}\check{\boldsymbol{u}}^{n-1} \cdot \boldsymbol{\nabla}\boldsymbol{q}^{n-1} \\ \widehat{\boldsymbol{q}}_2^{n} &= \boldsymbol{q}^{n-1} - \Delta t\check{\boldsymbol{u}}^{n-1/2} \cdot \boldsymbol{\nabla}\widehat{\boldsymbol{q}}_2^{n-1/2} \\ \widehat{\boldsymbol{q}}_2^{n+1/2} &= \widehat{\boldsymbol{q}}_2^{n} - \frac{\Delta t}{2}\check{\boldsymbol{u}}^{n} \cdot \boldsymbol{\nabla}\widehat{\boldsymbol{q}}_2^{n} \\ \widehat{\boldsymbol{q}}_2^{n+1} &= \widehat{\boldsymbol{q}}_2^{n} - \Delta t\check{\boldsymbol{u}}^{n+1/2} \cdot \boldsymbol{\nabla}\widehat{\boldsymbol{q}}_2^{n+1/2} \end{aligned} \tag{32}$$

where $\widetilde{\boldsymbol{q}}^n = \widehat{\boldsymbol{q}}_1^{n+1}$ and $\widetilde{\boldsymbol{q}}^{n-1} = \widehat{\boldsymbol{q}}_2^{n+1}$. In addition, the velocity field is defined as $\check{\boldsymbol{u}}^{n+1/2} = \boldsymbol{u}(x(t^{n+1}), t^{n+1/2})$ and is interpolated/extrapolated from known time levels but the difference from the SL method is that this velocity is now Eulerian since it is defined at the arrival point $x(t^{n+1})$. Note that the velocity field $\check{\boldsymbol{u}}$ is still somewhat akin to a velocity along characteristics since it is not updated in the RK2 method - rather it is interpolated (for n-1/2) and extrapolated (for n+1/2) using $\boldsymbol{u}^{n-1}$ and $\boldsymbol{u}^n$. Without this key step the OIFS method would not be able to retain its temporal accuracy.

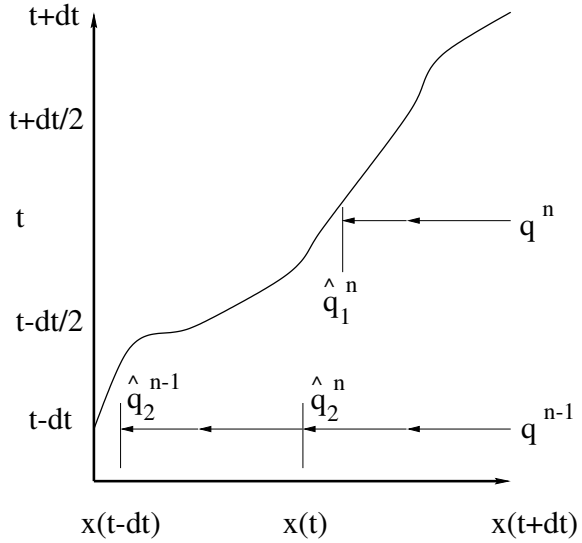


Figure 2: Schematic of the operator-integration-factor splitting (OIFS) trajectory computation. The thick line is the actual trajectory while the arrows denote the paths followed to compute the departure point values via a RK2 approximation.

However, if this were the complete story then OIFS would be rather limited. One can increase the maximum allowable time-step without losing accuracy or stability in two ways: the first is to use higher order RK methods such as RK3 and RK4 which have larger stability regions than RK2; the second is to use substepping such that each time-step is broken up into numerous smaller time-steps which are then passed to the RK method. Furthermore, these two points can be combined for further increases in the time-step. Clearly there is an efficiency price to be paid but if the implicit part of the solution is very costly then it makes sense to minimize the number of implicit solves by using a very large time-step thereby using multiple substeps in the OIFS method.

A few remarks regarding the OIFS method are now in order:

1. The success of the method relies on solving an advection problem, Eq. (30), efficiently.

2. For transport models with multiple tracers one must solve an OIFS problem for each individual tracer which results in a prohibitive cost.

3. No grid structure is required thereby OIFS can be used just as efficiently on completely unstructured grids.

4. On a distributed-memory computer the OIFS will scale because the advection operator is computed mostly on-processor; the only off-processor computation is in the communication required by the spatial discretization method to either compute derivatives (as in finite element, finite difference, or spectral methods) or fluxes (as in finite volumes, penalty, and discontinuous Galerkin methods).

5. Because the departure point values rely not on interpolation as in the SL method but on solving an advection operator, monotonicity can only be introduced by selecting spatial discretization methods which are monotonicity preserving.

While the OIFS method can be applied in conjunction with any spatial discretization method (that is, a method used to approximate derivatives such as the gradient in Eq. (30)) item 1 says that some spatial discretization methods may be impractical to use in an OIFS approach. For example, if high-order finite elements are used which result in a large sparse mass matrix then this matrix would have to be inverted at every step of the OIFS approach. Thus for an RK2 method, Eqs. (31) and (32), one would have to invert the mass matrix 6 times; however if two substeps are used then this number increases to 12! Clearly, spatial discretization methods which either do not have a mass matrix or have one that is diagonal is critical.

Item 2 says that the OIFS approach is really only practical for equation sets with few 'solution variables. Examples include incompressible Navier-Stokes, and hydrostatic and non-hydrostatic atmospheric and oceanic equations, to name a few relevant equation sets. Transport models probably are not a good choice for OIFS.

Item 3 shows the clear advantage of OIFS over SL on todays cache-based distributed-memory computer systems. Each stage of the RK advection problem will require some form of communication across processors but unlike in the SL method load balancing is guaranteed since each processor will always perform the same amount of work.

Item 4 may be viewed as either a virtue or a vice depending on how flexible the model is with regard to changing the spatial discretization methods. For example if one is using continuous numerical methods (such as finite difference, finite elements, or spectral methods) then it will be quite difficult to ensure monotonicity of the solution unless some other mechanism is applied (such as flux-corrected transport or some variant thereof) to eliminate oscillations and over/under shoots at the base of the discontinuity. However, if the discretization lends itself to be altered quite easily then one can change the spatial discretization for the advection problem to one which can ensure monotonicity and then switch back to the original method for the implicit solve. For example one could use finite/spectral elements for the implicit solve and discontinuous Galerkin (DG) for the advection problem [20, 8]. The advantage of using DG methods for the advection step is that one can then incorporate Riemann solvers or upwinding flux functions with total variation diminishing (TVD) schemes to eliminate any over/under shoots and spurious oscillations in the vicinity of shocks (see [21, 22, 23]).

4.3.3 Similarity between OIFS and SL Methods

Comparing the SL and OIFS methods one can see the strong similarities between the two methods. To compare the two methods we only need to look at the 1D advection equation which then simplifies to the following Lagrangian solution

$$q^{n+1} = q(\widetilde{x}^n, t^n). \tag{33}$$

Taking Eq. (28) and expanding in a Taylor series about the arrival point, x^{n+1}, we get for the velocity

$$u(\widetilde{x}^{n+1/2}, t^{n+1/2}) = u^{n+1/2} - \frac{\Delta t}{2} u^n u_x^{n+1/2} + O(\Delta t^2) \tag{34}$$

where $u^{n+1/2} = u(x^{n+1}, t^{n+1/2})$. The departure point can now be written as follows

$$\widetilde{x}^n = x^{n+1} - \Delta t u^{n+1/2} + \frac{\Delta t^2}{2} u^n u_x^{n+1/2} + O(\Delta t^3). \tag{35}$$

Incorporating this expansion for the departure point value yields

$$\widetilde{q}^n = q^n - \Delta t u^{n+1/2} q_x^n + \frac{\Delta t^2}{2} \left(u^n u_x^{n+1/2} q_x^n + u^{n+1/2} u^{n+1/2} q_{xx}^n \right) + O(\Delta t^3). \tag{36}$$

In contrast, a straight application of the OIFS RK2 method given in Eq. (31) yields

$$q^{n+1} = q^n - \Delta t u^{n+1/2} \frac{\partial}{\partial x} \left(q^n - \frac{\Delta t}{2} u^n q_x^n \right) \tag{37}$$

which, after differentiating and grouping terms, yields

$$\widetilde{q}^n = q^n - \Delta t u^{n+1/2} q_x^n + \frac{\Delta t^2}{2} \left(u^{n+1/2} u_x^n q_x^n + u^{n+1/2} u^n q_{xx}^n \right) + O(\Delta t^3). \tag{38}$$

Comparing Eqs. (36) with (38) shows that there is little difference between the SL and OIFS methods provided that the same time-integrator used for the trajectory computation of the SL method is used for the advection problem of the OIFS method. It should be kept in mind that the Taylor series expansion used in the SL derivation is only an approximation since in reality the SL method uses interpolation to obtain the departure point values. Therefore, it is not immediately obvious that any useful information can be obtained by such a simple analysis. However, it turns out that numerically both the OIFS and SL methods behave similarly provided that the same time-integrators are used. We have confirmed this for RK2, RK3, and RK4. To be clear, the difference in accuracy between the two methods is minimal. The main difference is in the stability regions of the two methods; the SL method, being a true Lagrangian method, avoids the CFL condition and thereby can use far larger time-steps than OIFS. In the next section we compare the results of both methods.

5 Results

For the numerical experiments, we use the normalized L_2 error norm

$$\|\phi\|_{L_2} = \sqrt{\frac{\int_\Omega (\phi_{\text{exact}} - \phi)^2 d\Omega}{\int_\Omega \phi_{\text{exact}}^2 \, d\Omega}}$$

to judge the accuracy of the methods. To compute the Courant number the elements are decomposed into their high-order (HO) grid points and these grid points form a fine grid which we refer to as the HO cells. The velocities and grid spacings are then defined at the centers of these cells. Using these definitions the Courant number is then defined as

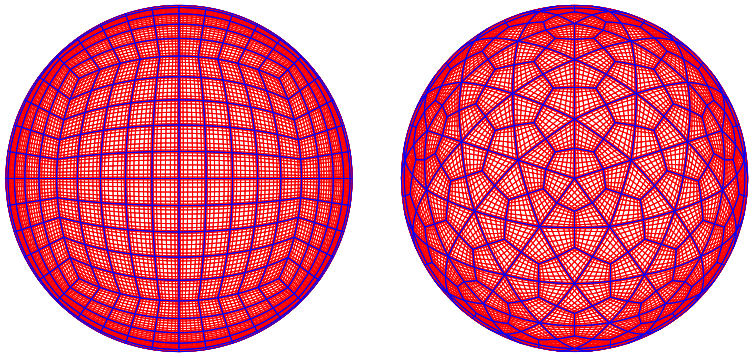
$$\mathrm{C} = \max \left(\frac{\lambda \Delta t}{\Delta s} \right)_{HO}^{e} \quad \forall e \in [1, ..., N_e]$$

where

$$\lambda = \begin{cases} U & \text{for} \quad \text{case 1} \\ U + \sqrt{\phi} & \text{for} \quad \text{all other cases} \end{cases}$$

where λ is the characteristic speed, $U = \sqrt{\boldsymbol{u} \cdot \boldsymbol{u}}$, and $\Delta s = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ is the grid spacing.

Four test cases are used to judge the performance of the HELSI time-integrators. Two types of grids used are shown in Fig. 3. Figure 3a shows the hexahedral grid which is a structured grid and Fig. 3b shows the icosahedral grid which is an unstructured grid; both grids have approximately the same number of points and both use 8th order polynomials. We use the hexahedral grid almost exclusively in this paper because it is a structured grid and, therefore, is quite easy to construct efficient search algorithms (see [9]) so that the SL method can be used efficiently (for search algorithms on icosahedral grids see [24]). However, the OIFS method does not care what the grid looks like and so either the hexahedral or the icosahedral grid can be used just as efficiently. For all of the problems studied here, however, the hexahedral grid yields more accurate results because the grid is aligned with the principle direction of the flow (east to west).

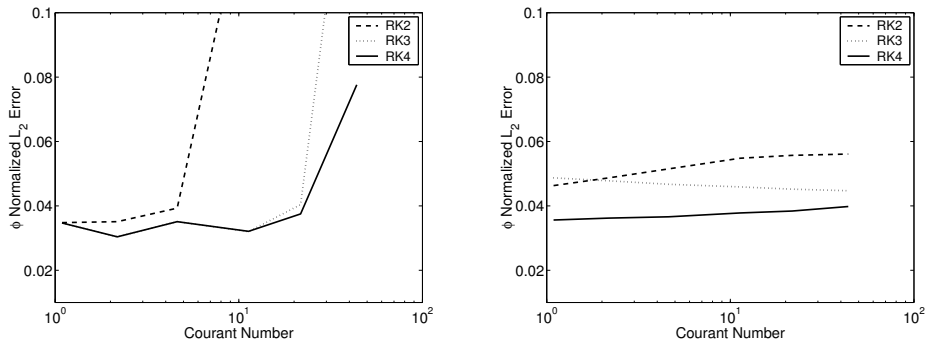


a) b)

Figure 3: The a) hexahedral and b) icosahedral grids with 38,402 and 34,562 grid points respectively.

5.1 Case 1: Passive Advection of a Cosine Wave

Case 1 concerns the solid body rotation of a cosine wave. The velocity field remains unchanged throughout the computation. This test case is integrated for 12 days which corresponds to one complete revolution of the wave (see [25] for its definition).



a) b)

Figure 4: Case 1. The ϕ L_2 error norm as a function of Courant number for a grid consisting of 6 elements each of 16th order for the a) SL and b) OIFS methods.

Figure 4 shows the errors for a 12 day integration for 6 elements tiling the sphere each of

which is 16th order accurate for a total of 1538 grid points (for description of the spectral element discretization see [9]) . In Fig. 4a we see that the SL method with RK4 yields better results than RK2 and RK3 for increasing time-step. Above Courant numbers of 4 the SL RK2 method loses accuracy quite rapidly. In contrast, SL RK3 retains its accuracy up to Courant numbers of 11 at which point it too loses its accuracy exponentially. Finally, SL RK4 does quite well even for Courant numbers as large as 50; however, for such large Courant numbers the SL method begins to lose its accuracy and should therefore not be used for this range of Courant numbers for this particular test case.

Figure 4b shows the results for OIFS with RK2, RK3, and RK4. The OIFS RK4 gives the best results but OIFS RK3 yields very comparable results. The OIFS RK2 results are not as good as those for RK3 and RK4. However, the one striking difference between the various OIFS and SL methods is that all the OIFS methods give somewhat similar results (of the same order of magnitude). The reason for this is simple; for an advection operator one can get the OIFS method to yield a solution for any size time-step. All one needs to do is increase the number of substeps per time-step. Thus in the results shown here the OIFS RK2 was able to run with Courant numbers as large as 50 but by using hundreds of substeps. Because one can increase the number of substeps then OIFS can use very large time-steps without incurring too large a loss in accuracy. However, as we will see in the next section using numerous substeps will penalize the overall efficiency of the method and therefore should be minimized whenever possible.

5.2 Case 2: Steady-State Nonlinear Zonal Geostrophic Flow

This case is a steady-state solution to the nonlinear shallow water equations. The equations are geostrophically balanced and remain so for the duration of the integration where the velocity field remains constant throughout the computation. The geopotential height ϕ undergoes a solid body rotation but since the initial height field is given as a constant band in the zonal direction (left to right) and the flow field is purely zonal, then the solution remains unchanged throughout the time-integration. The velocity field is the same as that used in case 1 and we integrate this test for 5 days (see [25] for its definition).

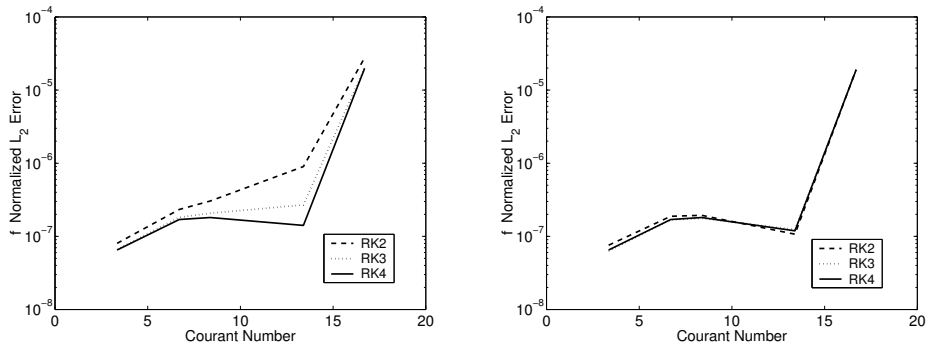


Figure 5: Case 2. The ϕ L_2 error norm as a function of Courant number for a grid consisting of 384 elements each of 8th order for the a) SL and b) OIFS methods.

Figure 5 shows the errors after a 5 day integration for 384 elements tiling the sphere each of which is 8th order accurate for a total of 24,578 grid points. In Fig. 5a we see that the SL method with RK4, once again, yields better results than RK2 and RK3 for increasing time-step. However, above Courant numbers of 15 all three SL methods lose their accuracy and yield similar results.

In contrast, Fig. 5b shows that all three OIFS methods yield very similar results for all Courant number values. In fact, comparing Figs. 5a and 5b one can see that all three OIFS methods yield

results quite similar to the SL RK4 method. Again, it is not suprising that the OIFS methods give similar results since it allows for substepping. In other words, the OIFS RK2 has a smaller stability region than the OIFS RK3 and RK4 methods. Thus, in order to use OIFS RK2 with a Courant number of 15 requires 8 substeps per time-step. Clearly, the number of substeps used will impact the overall efficiency of the scheme.

One final comment on the results presented in Fig. 5. Falconi and Ferretti [26] showed that the error of the semi-Lagrangian method is of the order

$$O\left(\Delta t^K + \frac{\Delta x^{N+1}}{\Delta t}\right) \tag{39}$$

where K and N represent the order of the trajectory approximation and the order of the interpolation functions, respectively. In other words for the trajectory equation, Eq. (27), K represents the order of accuracy of the numerical scheme used to solve this equation while N represents the order of the interpolation functions used to compute the variables at the feet of the characteristics (i.e., departure points). The importance of this analysis is that it then implies that for the SL method there exists an optimal time-step for a given grid spacing. In Fig. 5a we see that there is indeed an optimal time-step and it corresponds to a Courant number of approximately 13. Looking now at Fig. 5b we see that the optimal Courant number for the OIFS method is, once again, 13. In general we do not expect the optimal time-steps of both the SL and OIFS methods to coincide; however, what is interesting is that the OIFS method also seems to follow the convergence pattern given by Eq. (5) where N now represents the order of accuracy of the discrete spatial operators (such as the gradient operator required in the advection problem) and K the order of the time-integrator used to solve the advection step. Let us now turn to the computational cost of the SL and OIFS methods.

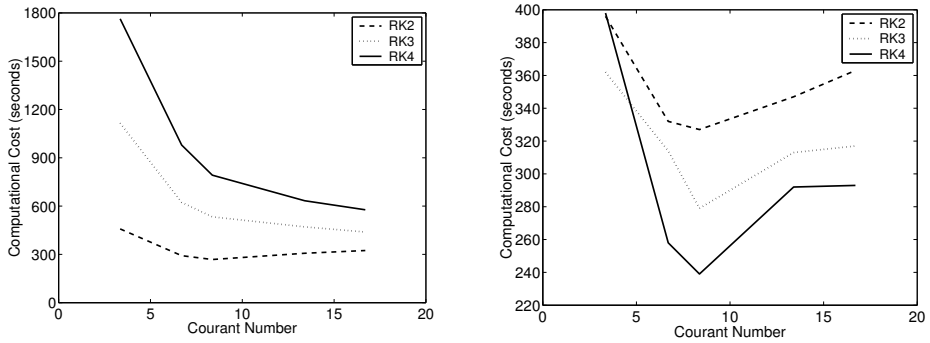


Figure 6: Case 2. The computational cost in seconds as a function of Courant number for a grid consisting of 384 elements each of 8th order for the a) SL and b) OIFS methods.

Figure 6 shows the computational costs associated with the results shown in Fig. 5 for the SL and OIFS methods. Not suprisingly, Fig. 6a shows that the SL RK4 method is the most costly. However, the cost of all three SL methods drops with increasing time-step. This too is to be expected since the cost of the SL method per time-step is constant regardless whether the time-step is small or very large. Thus for a pure advection problem, the cost would continue to decrease for increasing time-step. The reason why the cost plateaus is because the number of iterations required by the implicit solver to converge increases with increasing time-step.

Figure 6b, on the other hand, shows that the OIFS RK4 actually costs less even though it is higher order than RK2 and RK3. Recall, that the only reason why OIFS RK2 and RK3 can use Courant numbers nearing 15 is because substepping is used. The higher efficiency of OIFS RK4 is due to the fact that the RK4 method requires fewer substeps to be taken. For example, for the largest Courant number shown (17) the OIFS RK2 required 8 substeps while the OIFS RK3 required

only 3 substeps, and OIFS RK4 only 2. An interesting difference between the SL and OIFS methods is that the SL method, generally, becomes more efficient as the time-step is increased whereas the OIFS method clearly seems to have an optimal time-step range. For this specific test case the range seems to be somewhere around a Courant number of 7. However, as the time-step is increased the efficiency of SL RK2 begins to approach that of the OIFS methods. In short, SL RK2 and OIFS RK4 are the best compromises between accuracy and efficiency; we compare these two methods in Fig. 7. These are the two methods which we shall study in depth in the next two sections.

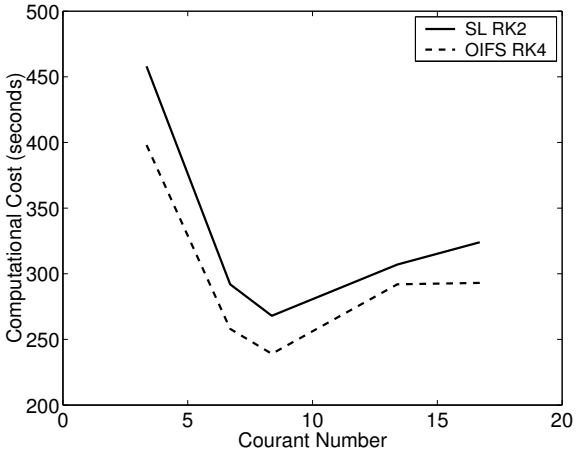


Figure 7: Case 2. The computational cost in seconds as a function of Courant number for a grid consisting of 384 elements each of 8th order for the SL RK2 (solid line) and the OIFS RK4 (dashed line) methods.

5.3 Case 3: Zonal Flow over an Isolated Mountain

This case uses the same initial conditions as case 2 with the addition of a conical mountain at $(\lambda, \varphi) = (180, 30)$. Due to the zonal flow impinging on the mountain, various wave structures form and propagate throughout the sphere. This test is run for 10 days (see [25] for details). Figure 8 shows the computational cost for a 10 day integration for 384 elements tiling the sphere each of which is 8th order accurate for a total of 24,578 grid points.

Figure 8 compares the computational cost of SL RK2 and OIFS RK4. Note that the efficiency plot of both methods looks somewhat erratic but as we shall see the results make perfect sense. For SL RK2 (solid line) the advection step cost remains constant regardless of the time-step size; this explains the decrease in computational cost for increasing Courant number. However, observe that the cost associated with C=12 increases. This is due to a very large jump in the number of iterations required by the iterative solver for convergence. For C=10 only 68 iterations are required whereas for C=12 that number jumps to 89. For the largest Courant number used (C=19) that number further increases to 141 iterations which explains why the cost increases.

For the OIFS RK4 (dashed line) there is a balance between the Courant number, the number of iterations, and the number of substeps. For $C < 10$ the OIFS RK4 only requires one substep; however, beyond this Courant number it requires 2 substeps. This is the reason why the cost increases for C=12 because the time-step has not increased by much between C=10 and C=12 but now twice the work has to be done for the advection step, not to mention that the number of iterations has also increased from 70 (for C=10) to 89 (for C=12).

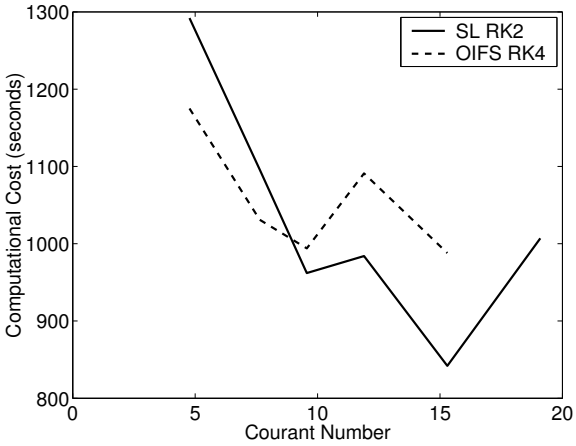


Figure 8: Case 3. The computational cost in seconds as a function of Courant number for a grid consisting of 384 elements each of 8th order for the SL RK2 (solid line) and the OIFS RK4 (dashed line) methods.

5.4 Case 4: Balanced Zonal Jet

This test case consists of a zonal jet and an unperturbed balanced initial geopotential height field. The balanced initial field should be maintained indefinitely but Galewsky et al. [27] suggest running the case for 5 days. This is a rather stringent test of shallow water models because if the accuracy and/or the resolution is not sufficiently high then the model will not be able to sustain the balanced initial field and the error will increase quite rapidly. In addition, because the jet is zonally positioned, then any grid that is not aligned with the zonal direction will have much more difficulty maintaining the jet.

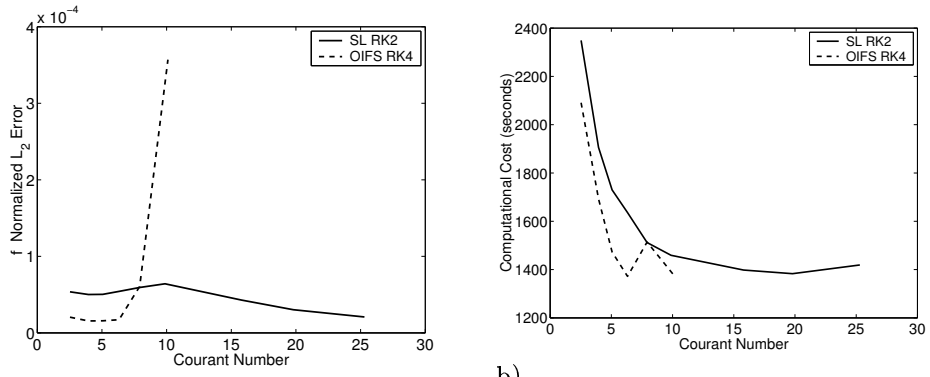


Figure 9: Case 4. The a) ϕ L_2 error norm and b) computational cost in seconds as a function of Courant number for the SL RK2 and OIFS RK4 methods using a grid consisting of 600 elements each of 8th order.

Figure 9 shows the errors and computational cost for a 5 day integration for 600 elements tiling the sphere each of which is 8th order accurate for a total of 38,402 grid points. Figure 9a shows that the SL method can sustain high accuracy for extremely large time-steps. In fact, for this test case

the SL method maintains high accuracy for Courant numbers of 25. In contrast, the OIFS method yields high accuracy for Courant numbers below 7. As the Courant number is increased the OIFS method cannot compete with the SL method in terms of accuracy for such large time-steps.

Figure 9b, however, shows that the OIFS method even with moderately large Courant numbers (C=7) is more efficient than the SL method using Courant numbers of 25. For Courant numbers below 7 both methods only require 38 iterations per time-step for the GMRES solver; however, the SL method using a Courant number of 25 requires over 200 iterations! Even though the SL method is taking a large time-step, this large time-step is affecting the condition number of the matrix which then requires a signficant amount of iterations. It should be mentioned that implementing state-of-the-art preconditioners may reduce the number of iterations; this should be further studied but is beyond the scope of the current work.

To show one of the advantages of the OIFS method we now run the simulation on an icosahedral grid. Recall that this grid represents an unstructured grid which poses no difficulties to the OIFS method; for the SL method one would have to devise a new search strategy and in order to do it efficiently requires exploiting the grid geometry. While this is not impossible to do for the SL method it does mean that whenever a new grid is used special care must be taken in order to construct efficient algorithms. It is also possible to construct generalized search algorithms but typically they are not as efficient as *ad hoc* methods. However, the point here is that the OIFS method needs no modification or optimization whatsoever whenever a new grid is introduced.

In Figs. 10 and 11 we show the geopotential height, ϕ, on the icosahedral and hexahedral grids. These results use 8th order polynomials and a Courant Number of 7 which, for this test case, is a factor of six larger than possible with the standard semi-implicit method. The icosahedral grid contains a total of 34,562 grid points while the hexahedral grid contains 38,402. The right panels of these figures show that the zonal jet is maintained for the duration of the 5 day integration.

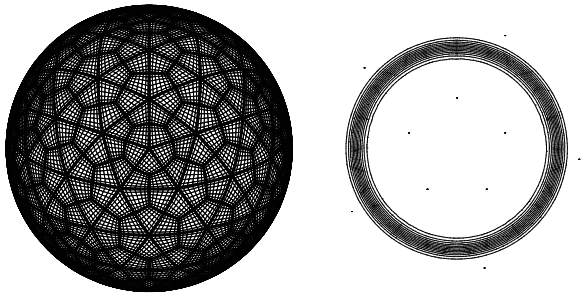


Figure 10: Case 4. The icosahedral grid (left panel) and ϕ contours (right panel) for a 5 day simulation. The simulation uses 8th order polynomials and a Courant Number of 7; the view is looking down on the North Pole.

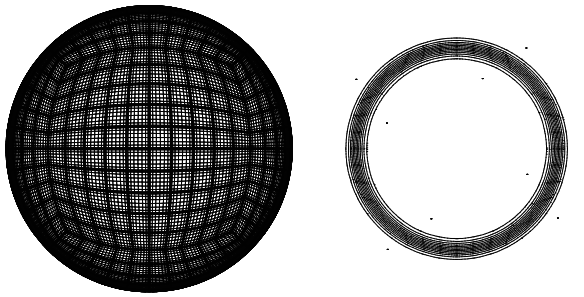


Figure 11: Case 4. The hexahedral grid (left panel) and ϕ contours (right panel) for a 5 day simulation. The simulation uses 8th order polynomials and a Courant Number of 7; the view is looking down on the North Pole.

6 Conclusions

A hybrid Eulerian-Lagrangian semi-implicit (HELSI) time-integrator has been presented. HELSI essentially builds the semi-Lagrangian (SL) and operator-integration-factor splitting (OIFS) method into the standard formulation of the semi-implicit method. This approach makes it very simple to augment a standard Eulerian semi-implicit method to a Lagrangian formulation; this increases the efficiency of the code by allowing the model to use very large time-steps. Furthermore, the construction of HELSI unifies the SL and OIFS methods into a single formulation. The advantage of this is that the similarity of the two methods can be appreciated and only a single computer code needs to be maintained. The SL and OIFS methods have their strengths and weaknesses which we have outlined and discussed. While the results shown here for the shallow water equations are quite impressive it remains to be seen whether such large time-steps can be taken for more complex equation sets such as the primitive atmospheric equations which we plan to address in future work.

Acknowledgments The author gratefully acknowledges the supported of the Office of Naval Research through program element PE-0602435N.

References

[1] B. Neta, F.X. Giraldo, and I.M. Navon, Analysis of the Turkel-Zwas scheme for the 2D shallow water equations in spherical coordinates, *Journal of Computational Physics* **133**, 102-122 (1997).

[2] E. Turkel and G. Zwas, Explicit large time-step schemes for the shallow water equations, in Advances in *Computer Methods for Partial Differential Equations*, edited by R. Vichnevetsky and R.S. Stepleman (IMACS, Lehigh University, 1979), p. 65.

[3] M. Kwizak and A.J. Robert, A semi-implicit scheme for grid point atmospheric models of the primitive equations, *Monthly Weather Review* **99**, 32-36 (1971).

[4] A. Robert, A semi-Lagrangian and semi-implicit numerical integration scheme for the primitive meteorological equations, *Journal of the Meteorological Society of Japan* **60**, 319-325 (1982).

[5] C. Temperton and A. Staniforth, An efficient two time-level semi-Lagrangian semi-implicit integration scheme, *Quarterly Journal of the Royal Meteorological Society* **113**, 1025-1039 (1987).

[6] Y. Maday, A.T. Patera, and E.M. Ronquist, An operator-integration-factor splitting method for time-dependent problems: application to incompressible fluid flow, *Journal of Scientific Computing* **5**, 263-292 (1990).

[7] J.P. Boyd, *Chebyshev and Fourier Spectral Methods* Dover Publications Inc, New York (2001) p. 265.

[8] D. Xiu, S.J. Sherwin, S. Dong, and G.E. Karniadakis, Strong and auxiliary forms of the semi-Lagrangian methods for incompressible flows, *Journal of Scientific Computing* in press (2005).

[9] F.X. Giraldo, J.B. Perot, and P.F. Fischer, A spectral element semi-Lagrangian (SESL) method for the spherical shallow water equations, *Journal of Computational Physics* **190**, 623-650 (2003).

[10] A. St-Cyr and S.J. Thomas, Nonlinear operator integration factor splitting for the shallow water equations, *Applied Numerical Mathematics* **52**, 429-448 (2005).

[11] F.X. Giraldo, Semi-Implicit Time-Integrators for a Scalable spectral element atmospheric model, *Quarterly Journal of the Royal Meteorological Society* **131**, 2431-2454 (2005).

[12] G.E. Karniadakis, M. Israeli, and S.A. Orszag, High-order splitting methods for the incompressible Navier-Stokes equations, *Journal of Computational Physics* **97**, 414-443 (1991).

[13] M.R. Kaazempur-Mofrad, P.D. Minev, and C.R. Ethier, A characteristic/finite element algorithm for time-dependent 3-D advection-dominated transport using unstructured grids, *Computer Methods in Applied Mechanics and Engineering* **192**, 1281-1298 (2003).

[14] A. Priestley, A quasi-conservative version of the semi-Lagrangian advection scheme, *Monthly Weather Review* **121**, 621-629 (1993).

[15] P. Garcia-Navarro and A. Priestley, A conservative and shape-preserving semi-Lagrangian method for the solution of the shallow-water equations *International Journal for Numerical Methods in Fluids* **18**, 273-294 (1994).

[16] F.X. Giraldo, The Lagrange-Galerkin spectral element method on unstructured quadrilateral grids, *Journal of Computational Physics* **147**, 114-146 (1998).

[17] D. Xiu and G.E. Karniadakis, A semi-Lagrangian high-order method for the Navier-Stokes equations, *Journal of Computational Physics* **172**, 658-684 (2001).

[18] J.P. Boyd, Prolate spheroidal wavefunctions as an alternative to Chebyshev or Legendre polynomials for spectral elements and pseudospectral algorithms. *Journal of Computational Physics* **199** 688-716 (2004).

[19] W.A. Mulder, Higher-order mass-lumped finite elements for the wave equation, *Journal of Computational Acoustics* **9**, 671-680 (2000).

[20] S.J. Sherwin, A substepping Navier-Stokes splitting scheme for spectral/hp element discretizations. In T. Matuson, A. Ecer, J. Periaux, N. Satufka, and P. Fox, Eds. *Parallel Computational Fluid Dynamics: New Frontiers and Multi-Disciplinary Applications* 43-52 North-Holland (2003).

[21] F.X. Giraldo, J.S. Hesthaven T. Warburton, Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations, *Journal of Computational Physics* **181**, 499-525 (2002).

[22] C. Eskilsson and S.J. Sherwin, A triangular spectral/hp discontinuous Galerkin method for modelling 2D shallow water equationsof the shallow-water equations *International Journal for Numerical Methods in Fluids* **45**, 605-623 (2004).

[23] F.X. Giraldo, High-order triangle-based discontinuous Galerkin methods for hyperbolic equations on a rotating sphere, *Journal of Computational Physics* in press (2005).

[24] F.X. Giraldo, Lagrange-Galerkin methods on spherical geodesic grids: the shallow water equations, *Journal of Computational Physics* **160**, 336-368 (2000).

[25] D.L. Williamson, J.B. Drake, J.J. Hack, R. Jakob, and P.N. Swarztrauber, A standard test set for numerical approximations to the shallow water equations in spherical geometry, *Journal of Computational Physics* **102**, 211-224 (1992).

[26] M. Falcone and R. Ferretti, Convergence analysis for a class of high-order semi-Lagrangian advection schemes, *SIAM Journal on Numerical Analysis* **35**, 909-940 (1998).

[27] J. Galewsky, R.K. Scott, and L.M. Polvani, An initial-value problem for testing numerical models of the global shallow water equations, *Tellus* **56**, 429-440 (2004).